Clutter reduction in Parallel Coordinates using locality-aware clustering*

Technical Report

1 Introduction

Parallel coordinates are a common way of presenting multivariate data in a single, two-dimensional visualization. Every datum is represented with a line passing over parallel coordinate axes corresponding to the data variables. The intention of parallel coordinates is to show a trend, either over time points or over another – natural or arbitrary – order of points.

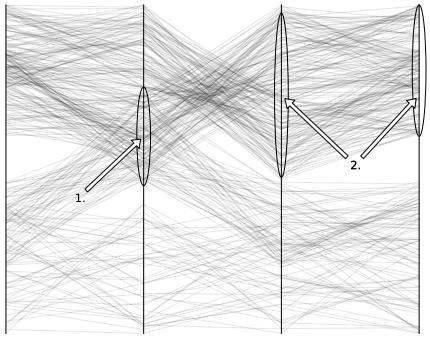
The usefulness of parallel coordinates is unfortunately seriously limited. Although it is intended to show a lot of data in multiple dimensions, these are exactly the circumstances to which it does not scale. With a lot of data, the visualization consists of a large number of overlapping lines, which makes the trends difficult to spot. With less data, any discovered patterns may be just random statistical flips.

The common solution to this problem is clustering of the data. This can be done either on the axis level, that is, by defining the clusters for each variable separately, on the level of the data set, by clustering data points. Both solutions contradict the initial purpose of the parallel coordinates. Clustering values by each variable separately ignores the fact that the variables are depicted on axes that are put one by another in a certain order. Clustering the data instances based on values of all variables suffers from the opposite problem: when putting two data instances in the same cluster, it measures their similarity across all variables although the visualization let us observe only relations that hold between pairs of variables.

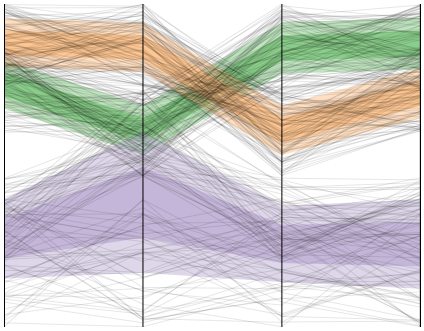
The situation is shown in Figure 1. Algorithms that cluster instances based on all variables might join the top most clusters (Region 2) and instead decide to split the bottom cluster. Region 1 would prove problematic for algorithms that cluster each variable separately. Cluster covering this region would be connected to a large interval of values on the neighboring variables, resulting in a bar that covers many instances and is therefore not informative.

We developed a clustering algorithm designed particularly for parallel coordinates. Its goal is to combine the properties of both general approaches described above. We need to cluster entire data instances, lest we lose the ability to track the instance over multiple variables and thus effectively construct multiple univariate visualizations. On the other hand, we want to reduce the clutter by having well-defined clusters for each individual variable.

*The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 318633



(a) Problematic areas



(b) Desired clusters

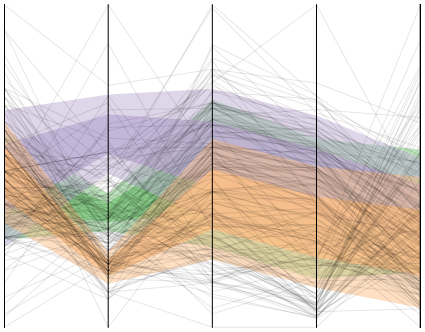
Figure 1: In the parallel coordinates plot (a), we would like to emphasize the clusters hidden in data (b). If we based the visualization of clusters on a method that clusters each variable separatelly, instances in region 1 would belong to the same cluster, although they belong to two different clusters. If the source of the visualization was a method that considers all variables simultaneously, simililar values of the top two clusters (region 2) would guide the the method towards a solution that would merge the top two clusters and instead split the botton cluster.

The objective function that corresponds to the above goal is

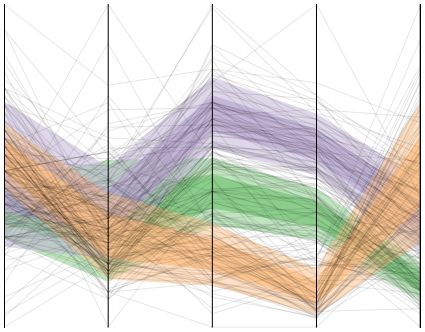
$$S = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\sqrt{\sigma_i}} \sum_{j=1}^{k} \sqrt{\sigma_{j,i}}, \tag{1}$$

where $\sigma_{j,i}$ denotes the variance of the i-th feature of the j-th cluster and σ_i stands for the overall variance of i-th feature.

Instead of optimizing this function directly, we developed a technique based on Gaussian Mixture Models learned using expectation maximization. The technique is described in the third section, and in the fourth we experimentally show that the resulting clusters are indeed better with respect to the above goal than those obtained by k-means clustering or standard Gaussian mixture models.



(a) K-means clusters



(b) LAC clusters

Figure 2: Comparison of clusters produced with k-means on the left (a) and locality aware clustering on the right (b) on wine dataset. Our method finds clusters that are narrower and better separated for each individual axis.

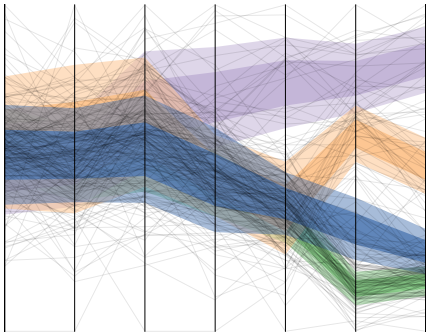
2 Related work

Several approaches have been proposed for reducing clutter. Most of them are based on aggregating data and later drawing the aggregates as densities [5], wavelets [10], polygons [1, 9] or bars with varying opacity [2, 4]. Clustering is a commonly used aggregating function [4, 7, 9, 1], usually combined with drawing cluster properties on the parallel coordinates plot.

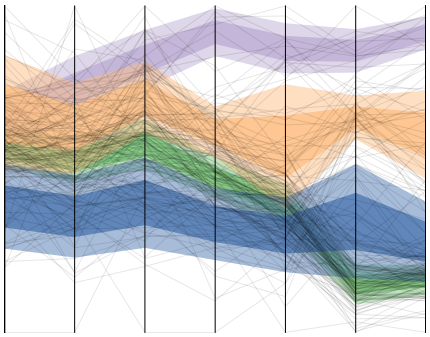
Fua et.al [4] use hierarchical clustering to provide multiple representations with different level of detail. Clusters are displayed as bands with opaque means and transparency linearly decreasing towards the cluster edge. This emphasizes the cluster width, but does not provide information about the instances within the cluster.

Johansson *et. al* [7] create high-precision textures for clusters and outliers and combine those with transfer function to create visualizations. Changing the transfer function allows the user to put emphasis on different aspects of data. Different statistical properties of the data can be displayed using visualization.

Novotny [9] uses a modified version of k-means clustering and displays clusters as polygons covering all examples in the cluster. Polygons are sorted by size before they are displayed to optimize the visibility of dense clusters.



(a) K-means clusters



(b) LAC clusters

Figure 3: Comparison of clusters produced with k-means on the left (a) and locality aware clustering on the right (b). Our method finds clusters that are narrower and better separated for each individual axis.

Andrienko et. al [1] propose two methods of visualizing distribution of attribute values within clusters. First approach splits intersection of cluster with each axis into quantiles and then draws lines connecting them, thus showing dense parts as narrow bands within the cluster. Another approach are ellipse plots that can be shown behind each axis and also visualize axis value distribution.

Our approach is similar to [4, 7, 9] as it also uses clusters to reduce clutter. Like [2], it uses probability based clustering that assigns example a probability of belonging to a specific cluster. Clusters are drawn with polygon like in [9, 1]. We borrow drawing quantiles in clusters from [1] to provide additional information about value distribution inside clusters.

3 Locality aware clustering

A Gaussian mixture model (GMM) is a parametric statistical model that assumes that the data originates from a weighted sum of several Gaussian sources. A GMM is defined by $p(x|\Theta) = \sum_{j=1}^{k} \alpha_j p(x|\theta_j)$, where α_j stands for the weight of the $j-th$ Gaussian source, θ_j represents sources parameters

initialize(μ, σ, ϕ)
for *step* **in** $1..max\_step$ **do**
  for each $i \leftarrow 1..m$, $j \leftarrow 1..k$ **do**
    $w_j^{(i)} \leftarrow p(z^{(i)} = j | x^{(i)}; \phi, \mu, \sigma)$
  end for

  $\phi_j \leftarrow \frac{1}{m} \sum_{i=1}^{m} w_j^{(i)}$
  $\mu_j \leftarrow \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}}$
  $\sigma_j \leftarrow \frac{\sum_{i=1}^{m} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}$
end for

Figure 4: Standard EM algorithm for estimating a mixture of Gaussian models

and k is the total number of Gaussian sources. Parameters of the GMM models are commonly estimated from unlabeled data using expectation maximization (Algorithm 4).

In our setup, we assume that the dataset X with m data instances x is generated by k Gaussian sources. Each source is described by its mean μ, covariance matrix *sigma* and prior ϕ. Prior is the probability probability that a data instance was generated by the selected source and is larger for clusters that cover more instances.

Estimating GMM parameters using EM finds a local optimum for the model parameters, so the choice of initial value of the model parameters is important. One may address this problem with multiple runs of the algorithm with different parameters or annealing [11]. In our experiments, we opted for starting the optimization with the best clustering produced by 10 runs of the k-means algorithm. The upside of this approach is that the resulting clusters are visually similar to the clusters produced by the k-means method, thus making the task of comparing them easier.

E-step of the optimization computes the weight matrix w containing probabilities that an instance belongs to the j-th cluster. This is done by evaluating the probability density function for normal distribution adjusted with the prior probability of the given cluster.

$$p(x, \mu, \sigma, \phi) = \phi \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

with the current model parameters.

In M-step, model parameters are updated with regard to the probabilities of instances belonging to the current cluster assignments. New value for *phi* is based on the normalized sum of probabilities of instances belonging to chosen cluster. New values for μ and σ are computed over the dataset, weighting each instance with the probability of it belonging to the given cluster.

Locality aware clustering (Algorithm 5) we developed is similar to GMM but takes the ordering of features into account. Features can be (re)ordered in any order in the visualization, but in the algorithm definition we will assume that the features appear in the same order in the dataset as they will appear on the visualization; features with consecutive indices are drawn on consecutive

$initialize(\mu, \sigma, \phi)$
for $step$ **in** $1..max\_step$ **do**
  for f **in** $features$ **do**
    $X' \leftarrow \{f_{i-\delta}, f_{i-\delta+1}, ..., f_{i+\delta}\}$
    for each $i \leftarrow 1..m$, $j \leftarrow 1..k$ **do**
      $w_j^{(i)} \leftarrow p(z^{(i)} = j | x'^{(i)}; \phi, \mu, \sigma)$
    end for

    $\phi_j \leftarrow \frac{1}{m} \sum_{i=1}^{m} w_j^{(i)}$
    $\mu_{j,f} \leftarrow \frac{\sum_{i=1}^{m} w_j^{(i)} x'^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}}$
    $\sigma_{j,f} \leftarrow \frac{\sum_{i=1}^{m} w_j^{(i)} (x'^{(i)} - \mu_j)(x'^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}$
  end for
end for

Figure 5: Locality aware clustering algorithm

axes.

Model updates are computed using a sliding window where each dimension of the model in updated in turn, but only data from neighboring dimensions $X' = select_dimension(X, d_{i-\delta}, ..., d_i, ..., d_{i+\delta})$ are used to compute cluster assignments and parameters updates for dimension i. This allows the algorithm to only consider values of the neighboring features when updating a single parameter dimension.

Windows size δ is a parameter to the algorithm we set to one. When windows size is set to 0, the algorithm is equivalent to fitting a 1D GMM model for each of the displayed features. When windows size is set to the number of dimensions, locality aware clustering is equivalent to the standard GMM clustering.

4 Evaluation

LAC was evaluated and compared with k-means clustering on multiple UCI datasets shown in table 1. Discrete features were removed from the dataset and missing values were replaces with feature averages.

Both methods were set to find 10 clusters in data and evaluated using the weighted sum of standard deviations:

$$S = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\sqrt{\sigma_i}} \sum_{j=1}^{k} \sqrt{\sigma_{j,i}}$$

Where $\sigma_{j,i}$ denotes the covariance of the i-th feature of the j-th cluster and σ_i stands for the overall covariance of i-th fetaure. Table 1 and figure 7 show the scores both methods achieved on UCI datasets.

Table 1: Sum of cluster std. deviations on UCI datasets (lower is better).

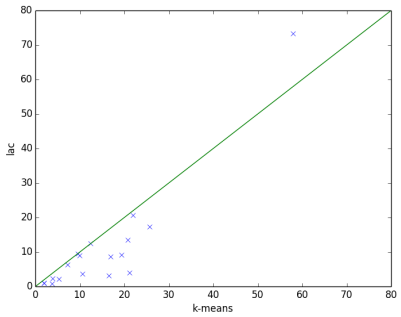
dataset	S(k-means)	S(gmm)	S(lac)
Adult	445.30	387.68	394.55
Anneal	20.51	34.27	13.53
Breast cancer Wisconsin	9.52	9.46	9.43
Brown	1.48	1.93	1.03
Bupa	23.78	23.88	20.66
Crx	19.21	21.61	9.17
Echocardiogram	10.63	12.16	3.64
Emotions	3.81	4.26	2.26
Glass	3.35	4.09	0.91
Heart disease	21.36	25.14	4.05
Horse colic	16.61	18.48	3.22
Housing	12.16	14.72	12.56
Imports-85	26.07	29.88	17.28
Ionosphere	4.73	4.29	2.18
Vehicle	17.01	17.66	8.75
Water treatment	57.06	61.36	73.36
WDBC	10.24	11.97	8.96
Wine	7.32	8.35	6.37
Yeast class RPR	1.94	1.93	1.03

5 Discussion

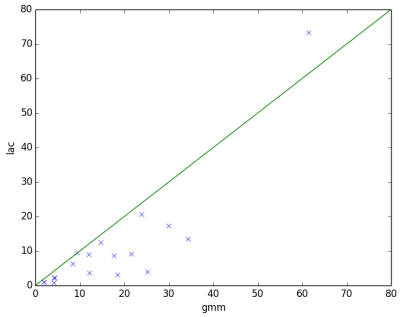
Results show that LAC outperforms k-means clustering on a majority of datasets, often by a wide margin. The dataset where it performs worse (by a large margin) contains data from different sensors in a water treatment plant, which might not be suitable for display in parallel coordinates.

Figures 2a, 2b, 3a and 3b show the difference between clusters produced. Clusters produced by LAC are generally better separated and therefore easier to visually track along the visualization.

As stated in section 3, LAC clustering comes with an inherited trade-off. Focusing on local trends of the data can lead to overlooking the global context and therefore not seeing the big picture. A simple example, where local clustering fails, is a dataset with multiple constant features (all examples have the same value). LAC uses a sliding window to update clusters and when this window slides over the constant features, the method looses the association of examples to clusters. When the window moves past the constant subspace, it will try to align clusters to examples that are not the same than before. Luckily, such patterns in data are easily discoverable (all lines crossing the axis in the same point). As such features are not informative, they can be removed from the projection, thus avoiding the weakness of the LAC clustering and resulting



(a) Score comparison with k-means



(b) Score comparison with gmm

Figure 6: Comparison of the score of k-means, gmm and lac clusters based on the table 1. Result for adult dataset has been removed for better clarity.

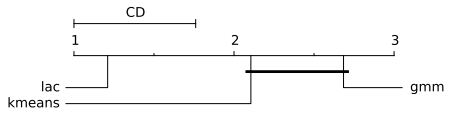


Figure 7: Comparison of ranks based on the table 1. For alpha=0.05, clusters produced with lac are significantly better than clusters produced with k-means/gmm.

in better clusters.

Alternative approach is to increase the windows size. With larger window size, algorithm will be more robus, but the resulting clusters will have more weight on the global trends.

Acknowledgments

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 318633

References

[1] G. Andrienko and N. Andrienko. Parallel coordinates for exploring properties of subsets. In *Procedings of the Second International Conference on Coordinated and Multiple Views in Exploratory Visualizations* (2004), pp. 93-104.

[2] M. Berthold and L. Hall. Visualizing fuzzy points in parallel coordinates. *IEEE Transactions on Fuzzy Systems* 11(3), (2005), pp. 369-374.

[3] A. P. Dempster, N.M. Laird, and D.B. Rubin. *Maximum likelihood from incomplete data via the EM algorithm.* JRSSB, 39:1-38, 1997

[4] Y. H. Fua, M. O. Ward and E. A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. In *Proceedings of IEEE Visualization* (1999), pp. 43-50.

[5] J. Heinrich and D. Weiskopf. Continuous parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 15(6), (2009) pp. 1531-1538.

[6] J. Johansson, P. Ljung and M. Cooper. Depth cues and density in temporal parallel coordinates. In *Proceedings of the Eurographics/IEEE-VGTC Symposium on Visualization* (2007), pp. 35-42.

[7] J. Johansson, P. Ljung, M. Jern and M. Cooper. Revealing structure within clustered parallel coordinates displays. In *Proceedings of the IEEE Symposium on Visualization* (2005), pp. 125-132.

[8] M. Novotny and H. Hauser Outlier-preserving Focus+Context visualization in parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5), (2006), pp. 893-900.

[9] M. Novotny. Visually effective information visualization of large data. In *Proceedings of the 8th Central European Seminar on Computer Graphics* (2004), pp. 41-48.

[10] R. Rosenbaum, J. Zhi and B. Hamann. Progressive parallel coordinates. In *Proceedings of the IEEE Pacific Visualization Symposium* (2012), pp. 25-32.

[11] N. Ueda, R. Nakano, Z. Ghahramani, and G. E. Hinton. SMEM algorithm for mixture models. *Neural computation* ,12(9), (2000) pp. 2109-2128.